

Nástroj pro tvorbu rozvrhů Rozvrhář

Dokumentace maturitní práce z programování

Šimon Vyhnis

# Zadání maturitní práce

Cílem práce je vytvořit v programovacím jazyce C# s použitím frameworku ASP.NET webovou aplikaci pro tvorbu školních rozvrhů. Aplikace uživateli na základě předem nahraných dat napovídá, které hodiny může do rozvrhu přidávat, aniž by s něčím kolidovaly. Do aplikace půjdou nahrávat data vyexportovaná ze systému Bakaláři. Data budou uložena v SQL databázi.

Teoretická část: Relační databáze

Prohlašuji, že jsem na práci pracoval samostatně pouze za pomoci použitých zdrojů a že v práci i v dokumentaci jasně vymezuji, které části kódů jsou mým originálním dílem, které jsou upravenou verzí a které jsou převzaty v plném rozsahu.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

podpis žáka

**Obsah**

[Zadání maturitní práce 2](#_Toc193920636)

[1. Teoretická část – Relační databáze 5](#_Toc193920637)

[1.1 Klíče a jejich typy 5](#_Toc193920638)

[1.2 Vazby mezi tabulkami 5](#_Toc193920639)

[1.3 Structured Query Language (SQL) 6](#_Toc193920640)

[1.4 Objektově relační mapování 7](#_Toc193920641)

[1.5 Výhody a nevýhody relační databáze 7](#_Toc193920642)

[2. Cíle práce 7](#_Toc193920643)

[3. Způsoby řešení a použité postupy 8](#_Toc193920644)

[3.1 Databázové schéma (návrh) 9](#_Toc193920645)

[4. Programovací prostředky 9](#_Toc193920646)

[5. Zhodnocení dosažených výsledků 9](#_Toc193920647)

[5.1 Míra splnění zadání 9](#_Toc193920648)

[5.2 Možnosti rozšíření 9](#_Toc193920649)

[6. Instalace 10](#_Toc193920650)

[6.1 Návod pro instalaci 10](#_Toc193920651)

[6.2 Systémové nároky 10](#_Toc193920652)

[7. Ovládání 11](#_Toc193920653)

[8. Seznam použitých zdrojů 12](#_Toc193920654)

[8.1 Seznam informačních zdrojů 12](#_Toc193920655)

[8.2 Seznam zdrojů kódu 13](#_Toc193920656)

# 1. Teoretická část – Relační databáze

Relační databáze jsou v současné době nejrozšířenějším způsobem ukládání dat ve větších informačních systémech. Základním prvkem relační databáze je relace – dvojrozměrná tabulka, která obsahuje jednotlivé záznamy (řádky). Každý sloupec v tabulce, atribut, má definovaný název a datový typ, data jednotlivých záznamů odpovídají struktuře definované atributy.

|  |  |  |  |
| --- | --- | --- | --- |
| id [int] | title [varchar 50] | name [varchar 200] | hours [int] |
| 1 | Mgr. | Monika Boušková | 20 |
| 2 | Ing. | Karel Kratochvíl | 15 |
| 3 | RnDr. | Stanislav Gottwald | 20 |

*Jednoduchý přiklad databázové tabulky učitelů. Tabulka má 4 atributy (id, titul, jméno a počet hodin) a obsahuje 3 záznamy (Moniku Bouškovou, Karla Kratochvíla a Stanislava Gottwalda).*

## 1.1 Klíče a jejich typy

Atributy nebo skupiny atributů mohou plnit funkci databázových klíčů. Základní typ je **kandidátní klíč**. Ten určuje, že daný atribut (nebo skupina) je pro každý záznam v jedné tabulce unikátní. Každá tabulka může obsahovat jeden nebo více kandidátních klíčů. Kandidátní klíče by měly obsahovat nejnižší možný počet atributů, který zaručí unikátnost.

Jeden z kandidátních klíčů může být **primární klíč**. Primárním klíčem může být pouze jeden atribut (nikoliv skupina) a slouží k identifikaci záznamu napříč tabulkami. Primární klíče se většinou generují jako speciální atribut přímo v databázi, a to buď jednoduše pořadím vytvořeného prvku, nebo pseudonáhodně a vygenerovaný atribut se většinou nazývá ID.

Ostatní kandidátní klíče mimo primárního se označují jako **alternativní**. Pro propojování tabulek v databázi mezi sebou se používají **cizí klíče**.

## 1.2 Vazby mezi tabulkami

Pro vytvoření složitějších datových struktur lze v relační databázi vytvářet vztahy mezi jednotlivými tabulkami. Vztah se tvoří pomocí primárního klíče jedné tabulky uloženého v jiné tabulce, tedy cizího klíče. Existují tři základní typy vztahů: 1:1, 1:n a n:n.

Ve vztahu **1:1** odpovídá každému záznamu z jedné tabulky právě jeden záznam v druhé tabulce. Obě tabulky obsahují cizí klíč.

|  |  |  |
| --- | --- | --- |
| id | title | teacher\_id |
| 1 | 1.A | 1 |
| 2 | 2.A | 3 |
| 3 | 3.A | 2 |

|  |  |  |
| --- | --- | --- |
| id | name | class\_id |
| 1 | Monika Boušková | 1 |
| 2 | Karel Kratochvíl | 3 |
| 3 | Stanislav Gottwald | 2 |

*Příkladem vztahu 1:1 může být např. vazba mezi třídou a třídním učitelem. Obě tabulky mají cizí klíč – teacher\_id a class\_id.*

Ve vztahu **1:n** odpovídá jednomu záznamu z jedné tabulky více záznamů z tabulky druhé. Cizí klíč obsahuje pouze druhá tabulka.

|  |  |  |
| --- | --- | --- |
| id | name | class\_id |
| 1 | Martin Novák | 1 |
| 2 | Petr Stuchlý | 1 |
| 3 | Martina Šikovná | 3 |

|  |  |
| --- | --- |
| id | title |
| 1 | 1.A |
| 2 | 2.A |
| 3 | 3.A |

*Příkladem vztahu 1:1 může být vazba mezi třídou a jejími studeny. Cizí klíč (class\_id) mají pouze studenti.*

Nejsložitějším typem vztahu je **n:n**, kde každému záznamu z jedné tabulky odpovídá více záznamů z druhé a zároveň každému záznamu z druhé tabulky může odpovídat více záznamů z první. Pro realizaci takového vztahu se používá spojovací tabulka, ve které se ukládají pouze vztahy záznamů mezi dvěma původními tabulkami.

|  |  |
| --- | --- |
| id | name |
| 1 | Martin Novák |
| 2 | Petr Stuchlý |
| 3 | Ema Šikovná |

|  |  |
| --- | --- |
| group\_id | student\_id |
| 1 | 1 |
| 3 | 1 |
| 3 | 3 |

|  |  |
| --- | --- |
| id | title |
| 1 | Angličtina 1 |
| 2 | Angličtina 2 |
| 3 | Němčina |

*Příkladem vztahu n:n může být vazba mezi studenty a skupinami. Vazby probíhají pomocí tabulky student\_to \_group. Martin Novák je ve skupině angličtina1 a němčina a zároveň ve skupině němčina je také Ema Šikovná.*

## 1.3 Structured Query Language (SQL)

Valná většina relačních databází používá pro vnější přístup standardizovaný jazyk Structured Query Language, který umožňuje načítání a úpravu dat pomocí příkazů (query). Příkazy vychází z angličtiny, ale jejich formát je pevně daný. Dělí se na 3 základní skupiny:

**Data definition language (DDL)** slouží k vytváření úpravě struktury pro samotná data. Patří sem především příkazy pro vytváření, úpravu a mazání tabulek (CREATE TABLE, ALTER TABLE, DROP TABLE) nebo práci s klíči.

**Data manipulation language (DML)** zastřešuje nejpoužívanější příkazy, protože zprostředkovává manipulaci se samotnými daty. Jak jejich načítání (příkaz SELECT), tak úpravu (příkazy INSERT, UPDATE DELETE).

**Data control language (DCL)** hraje důležitou roli především ve větších systémech, umožňuje vytvářet uživatele s různými právy a spouštět databázové transakce.

## 1.4 Objektově relační mapování

Pro přístup do relační databáze v objektově orientovaném programovaní se často využívá systému objektově relačního mapování. Tato technika umožňuje automatickou konverzi mezi databázovými záznamy a instancemi tříd a zajišťuje persistentní uchování dat v systému. Do velké míry také dokáže odstínit programátora od přímé práce s databází, kdy se databázové příkazy automaticky generují na pozadí.

## 1.5 Výhody a nevýhody relační databáze

Relační databáze má celou řadu výhod, asi nejdůležitější je vysoká míra organizovanosti a struktury v datech. Díky širokému rozšíření existuje mnoho různých implementací a je velká pravděpodobnost nalezení ideální implementace pro konkrétní aplikaci. Moderní relační databáze také poskytují zabezpečení a stabilitu potřebné pro ukládání citlivých dat.

Mezi nevýhody patří složitější návrh databázového schématu, náročnost na výkon při práci s velkými objemy dat a omezená flexibilita při změnách struktury. Také škálování relačních databází může být složitější ve srovnání s některými nerelačními řešeními.

# 2. Cíle práce

Vytváření rozvrhu na střední škole je poměrně složitý proces, do kterého vstupuje celá řada požadavků a proměnných. Přestože dnes většina škol má data o studentech a učitelích v informačních systémech, neposkytují tyto systémy efektivní nástroj pro vytváření rozvrhu. Cílem této maturitní práce je vytvořit samostatně běžící nástroj, který uživateli poskytne efektivní způsob vytváření rozvrhu.

Tento problém nabízí rámcově dvě možná řešení. První možností je generovat celý rozvrh na základě zadaných dat kompletně automaticky a následně nechat uživatele pouze upravovat vygenerovaný návrh. Druhou možností je uživateli pouze napovídat, které hodiny je možné do rozvrhu přidat, ale nechat samotné přidávání hodin na něm. Já jsem zvolil druhý přístup, protože není možné v rámci programu zpracovat všechny možné specifické požadavky pro rozvrh (zvlášť při zachování jednoduchého ovládání), a především není možné optimálně prioritizovat tyto požadavky a vyhodnocovat vhodné kompromisy. Program by měl tedy hlídat především kolize a specifické požadavky by měl zpracovávat uživatel.

Předem zřejmá limitace vytváření samostatného nástroje je nutnost zadat do programu velké množství dat před samotným začátkem tvorby rozvrhu. Tento problém se pokusím řešit možností importu dat z existujících systémů, ale jedná se spíše o zmírnění problému než řešení.

Pro využitelnost aplikace jsou podstatné její snadná dostupnost, uživatelská jednoduchost a spolehlivost. Snadnou dostupnost by měl zajistit vývoj aplikace jako webové, důležitá je optimalizace uživatelského rozhraní pro různé velikosti obrazovek. Jednoduchosti ovládání bych rád dosáhl využitím interaktivního vykreslování založeného na komponentech.

# 3. Způsoby řešení a použité postupy

## 3.1 Databázové schéma

A screenshot of a computer

AI-generated content may be incorrect.Základem celého programu jsou data, resp. jejich struktura v databázi. Na obrázku níže je můj návrh schématu před samotným vývojem programu. Výsledná databáze tomuto schématu téměř odpovídá, kromě tabulek out\_day a subject\_to\_room, protože měly sloužit pro funkcionality, které jsem nakonec neimplementoval.

Důležitým hlediskem, které není ve schématu znázorněno je možnost atributů nabývat hodnoty null. To je důležité především u cizích klíčů, respektive při mazání záznamů. Pokud totiž cizí klíč nemůže nabýt hodnoty null, dochází k tzv. kaskádovému mazání, tedy že při smazání záznamu, na který cizí klíč odkazuje, dojde rovněž ke smazání záznamu nesoucího cizí klíč. Toho využívám u předmětů a hodin. Nedává smysl, aby existoval předmět, který nemá učitele nebo skupinu studentů, protože by k němu v programu nebylo možné přistoupit, a proto je žádoucí ho při smazání skupiny, třídy, nebo učitele smazat také. Stejně tak u hodiny nedává smysl, aby existovala bez mateřského předmětu, tudíž je žádoucí ji kaskádovitě smazat. Naopak ke studentovi má uživatel přístup i mimo obrazovku třídy, a proto může zůstat bez třídy (atribut třídy se nastaví na hodnotu null a student zůstane zachován). Specifická je n:n vazba studentů a skupin, v ní sice při smazání studenta dochází ke kaskádovému smazání všech záznamů o jeho vztazích, ale samotné skupiny nejsou ovlivněny, stejně tak při mazání skupin.

## 3.2 Parsování CSV

Při přidávání studentů importem (ImportStudentsModal) program nejprve pomocí StreamReaderu přečte obsah souboru a poté ho zpracovává pomocí standartních metod pro prací s řetězci. Pro každého nalezeného studenta se snaží nalézt třídu odpovídající názvem. Před vložením dat do databáze se uživateli vypíše informace o tom, kolik uživatelů bylo nalezeno a pro kolik z nich se podařilo najít třídu, aby se minimalizovalo riziko přidání chybně načtených dat do databáze.

## 3.3 Modals

Pro větší přehlednost uživatelského rozhraní neobsahuje aplikace oddělená okna pro přidávání a mazání dat, ale řeší tuto problematiku pouze menšími vyskakovacími okny (modals). Tyto okna jsou zároveň řešená jako generické komponenty AddModal a DeleteModal, do kterých se pouze pomocí parametru vkládá konkrétní formulář pro daný objekt. Díky tomu není nutné zbytečně opakovaně psát kód pro obsluhu vyskakovacího okna.

## 3.4 Přihlašování

Pro přihlašování využívám funkcionality obsažené v ASP.NET core, které ověřují uživatele pomocí tokenu uloženém v souborech cookies. Zprostředkovávají také hashování a ukládání hesel. Stránky pro přihlašování a odhlašování jsou na rozdíl od ostatních obrazovek aplikace vytvořené jako oddělené stránky, ne jako komponenty, protože pro změnu cookies je třeba poslat nový request (komponenty komunikují pouze pomocí websocketu).

## 3.5 TableData

Algoritmicky asi nejzajímavější částí programu je třída TableData, která zprostředkovává data pro tabulku s rozvrhem a její úpravy. Kvůli většímu rozsahu jsem se rozhodl logiku v TableData oddělit do zvláštního servisu. Využívám pro něj dependency injection (DI), kdy TableData jednoduše přidám mezi servisy při spouštění programu (Program.cs) a následně jen získávám instanci „injektováním“ tam, kde data potřebuji.

### 3.5.1

# 4. Programovací prostředky

## 4.1 Programovací jazyk a framework

Program je napsán v programovacím jazyce C# a běží na frameworku ASP.NET core. C# poskytuje moderní přístup k asynchronnímu programování a řešení hodnot null, obě problematiky se v kódu hojně vyskytují. Framework ASP.NET, resp. Blazor, umožňuje vytváření tzv. razor komponent, ve kterých se kombinují html elementy s business logic v rámci jednoho souboru, což výrazně usnadňuje vývoj.

## 4.2 Databáze a objektově relační mapování

Program přistupuje k datům výhradně skrz Entity First framework, který umožňuje velmi pokročilé objektově relační mapování, včetně generování velmi složitých databázových příkazů. Jako databázi jsem původně chtěl použít Microsoft SQL Server, ale později jsem pro jednodušší lokální instalaci programu zvolil Sqlite.

# 5. Zhodnocení dosažených výsledků

## 5.1 Míra splnění zadání

Požadavky ze zadání byly téměř kompletně splněny. Aplikace využívá zadané programovací prostředky a napovídá uživateli, které hodiny může přidat. Navíc ukazuje, které hodiny ještě zbývá přidat a umožňuje sledování úvazků pedagogů. Studenty a učitele je možné nahrávat z CSV souborů, jež lze vyexportovat ze systému Bakaláři.

## 5.2 Možnosti rozšíření

Aplikace nabízí celou řadu možných rozšíření, protože je mnoho specifických požadavků při tvorbě rozvrhů, které nejsou implementovány. Se dvěma takovými funkcemi počítá původní databázové schéma. První z nich je možnost zadat do programu dny, ve kterých je nějaký učitel nepřítomný a nemohly by mu tedy být přidávány hodiny na tento den. Druhým je umístění některých předmětů pouze do určitých učeben (např. tělesná výchova pouze v tělocvičně, laboratoře pouze ve specializovaných učebnách apod.).

V aplikaci chybí možnosti pro export rozvrhu po jeho vytvoření. Mohl by být exportován jak pro zobrazení (např. pdf), tak pro nahrání do jiného programu. Systém bakaláři bohužel umožňuje import varianty rozvrhu pouze v interním formátu s lokálními klíči, tudíž není možné do něj hotový rozvrh importovat.

Uživatelské rozhraní také obsahuje prostor pro zlepšení, např. atraktivnější indikace stavu načítání, více animací nebo více nápovědy pro nezkušené uživatele. Celý program by také mohl být testován pomocí unit testů, což by mohlo zlepšit jeho spolehlivost, ale vyžadovalo by to použití velké míry mockování dat.

# 6. Instalace

## 6.1 Návod pro instalaci

Pro lokální spuštění je třeba extrahovat složku TimetableHelper.zip a poté spustit soubor TimetableHelper.exe, který se v ní nachází. Pro správné fungování programu je nutné, aby .exe soubor zůstal ve složce se všemi ostatními extrahovanými soubory, především složkou wwwroot.

A screen shot of a computer program

AI-generated content may be incorrect.Po spuštění souboru by se mělo otevřít konzolové okno, do nějž program vypíše, na jakém portu běží. Webová aplikace běží na vypsané adrese.

## 6.2 Systémové nároky

Hostování aplikace je možné na zařízeních s operačním systémem Windows 7, 8 a 10 a Windows Server, podmínkou je rozhraní .NET Framework verze 4.8 nebo novější. Za použití nástroje Nginx je možné hostování i na některých distribucích linux. Minimální hardware nároky jsou kapacita RAM 1 GB, frekvence procesoru 1kHz a dostupné 4 GB na disku.

Pro klientský přístup k aplikaci jsou podporované prohlížeče Google Chrome, Mozilla Firefox, Microsoft Edge a Apple Safari v nejnovějších verzích. Aplikace je optimalizovaná pro obrazovky o velikosti alespoň 1280x720.

# 7. Ovládání

# 8. Seznam použitých zdrojů

## 8.1 Seznam informačních zdrojů

Dokumentace Bootstrap: Tables. Upraveno 2023, cit. 15. 3. 2025. Dostupné online: <https://getbootstrap.com/docs/5.3/content/tables/>

Dokumentace Bootstrap: Select. Upraveno 2023, cit. 15. 3. 2025. Dostupné online: <https://getbootstrap.com/docs/5.3/forms/select/>

Dokumentace Bootstrap: Grid System. Upraveno 2023, cit. 15. 3. 2025. Dostupné online: <https://getbootstrap.com/docs/5.3/layout/grid/>

Dokumentace EF Core: Creating Model. Upraveno 2023, cit. 15. 3. 2025. Dostupné online: <https://learn.microsoft.com/ef/core/modeling/>

Dokumentace EF Core: Relationships. Upraveno 2023, cit. 15. 3. 2025. Dostupné online: <https://learn.microsoft.com/ef/core/modeling/relationships>

Dokumentace ASP.NET: Authentication. Upraveno 2024, cit. 20. 3. 2025. Dostupné online: <https://learn.microsoft.com/aspnet/core/blazor/security>

Wikipedia: Relational database. Upraveno 2025, cit. 25. 3. 2025. Dostupné online: <https://en.wikipedia.org/wiki/Relational_database>

## 8.2 Seznam zdrojů kódu

1. V rámci všech dialogových oken (Modals) byl využit upravený kód ze stránky Stack Overflow.  
   Stack Overflow: *“ How to use Bootstrap modal in Blazor client app?”* [online]. 2020, cit. 15.3.2025. Dostupné online: <https://stackoverflow.com/questions/59256798>
2. A screen shot of a computer program

   AI-generated content may be incorrect.V rámci animací pro úpravu buňky v tabulce rozvrhu (wwwroot/app.css) byl využit upravený CSS kód vygenerovaný za pomoci AI chatbotu ChatGPT ve verzi z 8. března 2025 s následující instrukcí: *“Write CSS for appearing white plus and green background in the middle of a table cell on hover, with faster disappearing”*. Vygenerovaný kód:
3. A computer code on a black background

   AI-generated content may be incorrect.Ve funkci models/Teacher.GetInitials() byl využit upravený kód vygenerovaný za pomoci AI chatbotu ChatGPT ve verzi z 15. března 2025 s následující instrukcí: *“ Write C# method, that turn string name into initial”*. Vygenerovaný kód: